Conversion Operators Solutions

Copyright © James Raynard 2021

Conversion Operator

- What is a conversion operator?
  - A conversion operator is a member function which converts an object to some other type
  - It does not modify the object, but returns a value of the desired type which corresponds in some way to the object's state
  - It is called when a variable of that type is expected, but an object of the class is provided instead
- Write a simple program that demonstrates a conversion operator

Copyright © James Raynard 2021

Implicit conversion

- What is meant by an implicit conversion?
  - An implicit conversion occurs when we do not directly ask for one
  - It occurs when the compiler recognises the conversion is needed to make the code compile
- Are implicit conversions always desirable?
  - They can be useful, for example in conditionals

```
if (obj)                    // Implicit conversion of obj to bool
  // Object is valid
```

  - However, they often occur when not expected, due to coding mistakes
  - Instead of a compiler error, we get code which behaves strangely

Copyright © James Raynard 2021

Explicit conversion

- What is meant by an explicit conversion?
  - An explicit conversion occurs when we directly ask for the object to be converted to a different type
  - Usually this is done by a static_cast
- It is possible to prevent implicit conversions?
  - In Modern C++, we can declare the conversion operator as explicit
  - In this case, the conversion will only be done if we ask for it
  - If we write code that requires an implicit conversion, we get a compiler error (with one exception!)
- Are there any exceptions to this?
  - Yes, if the conversion is to bool and occurs inside a conditional, an implicit conversion is allowed

Copyright © James Raynard 2021